KATHOLIEKE
UNIVERSITEIT
LEUVEN

DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

RESEARCH REPORT 9911

AN EXACT PROCEDURE FOR THE RESOURCE-CONSTRAINED WEIGHTED EARLINESS-TARDINESS PROJECT SCHEDULING PROBLEM

by
M.VANHOUCKE
E. DEMEULEMEESTER
W. HERROELEN

D/1999/2376/11

AN EXACT PROCEDURE FOR THE RESOURCE-CONSTRAINED WEIGHTED EARLINESS-TARDINESS PROJECT SCHEDULING PROBLEM

Mario VANHOUCKE • Erik DEMEULEMEESTER • Willy HERROELEN

February 1999

Operations Management Group
Department of Applied Economics
Katholieke Universiteit Leuven
Naamsestraat 69, B-3000 Leuven (Belgium)
Phones: 32-16-32 69 65, 32-16-32 69 72, 32-16-32 69 70, Fax 32-16-32 67 32
E-mail: <first name>.<name>@econ.kuleuven.ac.be

AN EXACT PROCEDURE FOR THE RESOURCE-CONSTRAINED WEIGHTED EARLINESS-TARDINESS PROJECT SCHEDULING PROBLEM

Mario VANHOUCKE • Erik DEMEULEMEESTER • Willy HERROELEN

ABSTRACT

In this paper we study the resource-constrained project scheduling problem with weighted earliness-tardiness penalty costs. Project activities are assumed to have a known deterministic due date, a unit earliness as well as a unit tardiness penalty cost and constant renewable resource requirements. The objective is to schedule the activities in order to minimize the total weighted earliness-tardiness penalty cost of the project subject to the finish-start precedence constraints and the constant renewable resource availability constraints. With these features the problem becomes highly attractive in just-in-time environments.

We introduce a depth-first branch-and-bound algorithm which makes use of extra precedence relations to resolve resource conflicts and relies on a fast recursive search algorithm for the unconstrained weighted earliness-tardiness problem to compute lower bounds. The procedure has been coded in Visual C++, version 4.0 under Windows NT and has been validated on a randomly generated problem set.

Keywords: *Resource-constrained project scheduling; Weighted earliness-tardiness costs; Branch-and-bound*

1. Introduction

Most of the work in project scheduling has focused on regular measures of performance. A regular measure of performance is a nondecreasing function of the activity completion times (in the case of a minimization problem), with the minimization of the project duration as the most popular one. Other examples are the minimization of the mean flowtime, the mean lateness, the mean tardiness and the percentage of jobs tardy.

In recent years scheduling problems with nonregular measures of performance have gained increasing attention. A nonregular measure of performance is a measure for which the above definition does not hold. A popular nonregular measure of performance in the literature is the maximization of the net present value (*npv*) of the project. In this case, a positive or negative cash flow is assigned to each activity and the objective is to schedule the activities in order to maximize the total net present value of the project. We can distinguish between procedures for the unconstrained max-*npv* project scheduling problem and those for the resource-constrained max-*npv* project scheduling problem. For an overview of the literature, we refer to Herroelen et al. (1997) and De Reyck and Herroelen (1998).

Another nonregular measure of performance, which is gaining attention in JIT environments, is the minimization of the weighted earliness-tardiness penalty costs of the activities in a project. In this problem, a due date, a unit earliness penalty cost and a unit tardiness penalty cost are assigned to the activities and the objective is to schedule the activities to minimize the weighted penalty cost of the project. This problem often occurs in practice since many project schedulers have to deal with due dates and penalty costs. Costs of earliness include extra storage requirements and idle times and implicitly incur opportunity costs. Tardiness leads to customer complaints, loss of reputation and profits, monetary penalties or goodwill damages. The problem is faced by many firms hiring subcontractors, maintenance crews as well as research teams. Recently, the unconstrained weighted earliness-tardiness project scheduling problem (denoted as $cpm|early/tardy$, according to the classification scheme of Herroelen et al. (1998), and subsequently denoted as *WETPSP*, i.e. the **W**eighted **E**arliness-**T**ardiness **P**roject **S**cheduling **P**roblem) has been optimally solved by the efficient exact recursive search algorithm developed by Vanhoucke et al. (1999).

In this paper we present a branch-and-bound algorithm to minimize the weighted earliness-tardiness penalty costs in project networks subject to zero-lag finish-start precedence constraints and renewable resource constraints ($m,1|cpm|early/tardy$, subsequently denoted as *RCPSPWET*, i.e. the **R**esource-**C**onstrained **P**roject **S**cheduling **P**roblem with **W**eighted **E**arliness-**T**ardiness costs). The *RCPSPWET* extends the NP-hard resource-constrained project scheduling problem under the minimum makespan objective (problem $m,1|cpm|C_{max}$) to the nonregular early/tardy performance measure. To the best of our knowledge, no exact procedure has yet been suggested for its solution. The solution procedure proposed in this paper computes lower bounds using the exact recursive search algorithm for the *WETPSP* of Vanhoucke et al. (1999). The branching strategy resolves resource conflicts through the addition of extra precedence relations based on the concept of minimal delaying alternatives developed by Demeulemeester and Herroelen (1992, 1997) and further explored by Icmeli and Erengüç (1996).

The organisation of the paper is as follows. In section 2 we present a conceptual formulation of the *RCPSPWET*. Section 3 describes the logic of the branch-and-bound algorithm. Section 4 illustrates the algorithm on a numerical example and in section 5 we report detailed computational results on a randomly generated problem set. In section 6 we give our overall conclusions.

2. The deterministic *RCPSPWET*

Assume an activity-on-the-node (AON) network $G=(N,A)$ where the set of nodes, N, represents activities and the set of arcs, A, represents finish-start precedence constraints with a time lag of zero. The activities are numbered from the dummy start activity 1 to the dummy end activity n. The duration of an activity is denoted by d_i ($1 \leq i \leq$ n) and its known deterministic due date by h_i. The completion time of activity i is denoted by the nonnegative integer variable f_i ($1 \leq i \leq$ n). The earliness of activity i can be computed as $E_i = \max(0, h_i - f_i)$ and its tardiness as $T_i = \max(0, f_i - h_i)$. If e_i and t_i respectively denote the per unit earliness and tardiness penalty cost of activity i, its total earliness-tardiness cost is $e_iE_i + t_iT_i$. There are K renewable resources with a_k ($1 \leq k \leq K$) as the availability of resource type k and with r_{ik} ($1 \leq i \leq$ n, $1 \leq k \leq K$) as the resource requirements of activity i with respect to resource type k. The *RCPSPWET* can be conceptually formulated as follows:

$$\text{Minimize} \sum_{i=2}^{n-1} (e_i E_i + t_i T_i) \qquad [1]$$

Subject to

$$f_i \leq f_j - d_j \qquad \forall (i,j) \in A \qquad [2]$$

$$E_i \geq h_i - f_i \qquad \forall i \in N \qquad [3]$$

$$T_i \geq f_i - h_i \qquad \forall i \in N \qquad [4]$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k \qquad k = 1,2,\ldots,K \qquad t = 1,2,\ldots,T \qquad [5]$$

$$f_1 = 0 \qquad [6]$$

$$f_i \in N, E_i \in N, T_i \in N \qquad i = 1,2,\ldots,n \qquad [7]$$

where $S(t)$ denotes the set of activities in progress in period $]t\text{-}1,t]$. The objective in Eq. 1 minimizes the total weighted earliness-tardiness cost of the project. The constraint set given in Eq. 2 maintains the finish-start precedence relations among the activities. The constraint sets in Eq. 3 and Eq. 4 compute the earliness and tardiness of each activity and Eq. 5 represents the renewable resource constraints. Eq. 6 forces the dummy start activity to end at time zero and Eq. 7 ensures that the activity finishing times as well as the earliness and tardiness assume nonnegative integer values.

3. The branch-and-bound algorithm

3.1 Description of the search tree and branching strategy

It is clear that the optimal solution to the *WETPSP* provides a lower bound on the corresponding *RCPSPWET*. We exploit this fact by computing, at the root of the branch-and-bound tree, an initial lower bound *lb* on the weighted earliness-tardiness cost using the recursive procedure of Vanhoucke et al. (1999). If this solution is resource feasible, we have the optimal solution for the *RCPSPWET* and the procedure terminates. If, however, a resource conflict can be detected, we branch into the next level of the branch-and-bound tree to generate a set of delaying alternatives. A *resource conflict* occurs when there is at least one period $]t\text{-}1,t]$ for which $\exists k \leq K$: $\sum_{i \in S(t)} r_{ik} \leq a_k$.

According to Demeulemeester and Herroelen (1992, 1997), it is sufficient to consider only a set *DS* of *minimal delaying alternatives* to resolve a resource conflict, i.e. *DS* contains minimal sets of activities which, when delayed, release enough resources to resolve the resource conflict and which do not contain any other delaying alternative as a subset. Each of these minimal delaying alternatives is delayed by each of the remaining activities in progress in period $]t^*\text{-}1,t]$ (the period of the *first* encountered resource conflict). Therefore, each minimal delaying alternative can give rise to several *minimal delaying modes* (De Reyck, 1998). For each delaying mode we impose additional precedence relations to resolve the resource conflict (Icmeli and Erengüç, 1996) and compute a new lower bound using the recursive search procedure. If, for example, a resource conflict is caused in period $]t^*\text{-}1,t]$ by the set of activities $S(t^*)=\{1,2,3\}$ and the delaying set contains two minimal delaying alternatives, i.e. $DS = \{\{1,2\},\{3\}\}$, then the three different delaying modes are ($1 \prec 3$), ($2 \prec 3$) and ($3 \prec 1,2$) corresponding to four additional precedence relations.

Each delaying mode corresponds to a *node* in the branch-and-bound tree which will be further explored during the branching process. We select among these nodes the delaying mode with the smallest *lb*. If the lower bound of a node corresponds to a solution which is resource feasible, we update the upper bound *ub* of the project (initially $ub = \infty$) and search for the following delaying mode at this level. If, however, the lower bound of a node is greater than or equal to the current upper bound, we fathom this node and select also the following delaying mode at this level. If the lower bound is smaller than the current upper bound, we generate a new set of delaying alternatives at the next level of the tree. If there are no delaying modes left, we backtrack and proceed in the same way at the previous level. The algorithm stops when we backtrack to the initial level of the branch-and-bound tree.

3.2 Node fathoming rules

Essentially, each node in the branch-and-bound tree represents the initial project network extended with a set of zero-lag finish-start precedence constraints to resolve resource conflicts. Therefore, it is possible that a certain node represents a project network which has been examined earlier at another node in the branch-and-bound tree. One way of checking whether two nodes represent the same project network is to check the added precedence constraints. If a node is encountered for which the set of added precedence constraints is *identical* to the set of precedence constraints associated with a previously examined node, the node can be fathomed. Moreover, the *subset dominance rule* developed by De Reyck (1998) for the resource-constrained project scheduling problem with generalized precedence relations also holds for the *RCPSPWET*, and can be applied when a node is compared to a previously examined node in *another path* in the branch-and-bound tree:

Subset dominance rule: *If the set of added precedence constraints which leads to the project network in node x contains as a subset another set of precedence constraints leading to the project network in a previously examined node y in another branch of the search tree, node x can be fathomed.*

Since the detection of a dominated subset is much faster than the calculation of a lower bound, we first check if the node can be dominated by a previously examined node. Only if the node can not be fathomed due to the subset dominance rule, we calculate a lower bound. Clearly, as mentioned earlier, a node can also be fathomed when its lower bound is greater than or equal to the current upper bound or when its solution is resource feasible. Of course, in the last case, the upper bound has to be updated first.

3.3 The algorithm

When AC_z denotes the set of added precedence constraints in node z (with respect to the original set of precedence constraints A) and x always denotes the current node in the branch-and-bound tree, the detailed steps of the branch-and-bound algorithm can be written as follows:

STEP 1. INITIALISATION

- Let $ub = \infty$ be the upper bound of the weighted earliness-tardiness cost.
- Initialize the level of the branch-and-bound tree: $p = 0$.
- Compute a lower bound lb on the weighted earliness-tardiness cost using the recursive solution procedure.
- If this solution is resource feasible, i.e. for each period $]t\text{-}1,t]$ and $\forall k \leq K$:

  $\sum_{i \in S(t)} r_{ik} \leq a_k$ STOP.
- Go to STEP 2.

STEP 2. MINIMAL DELAYING ALTERNATIVES

- Increase the level of the branch-and-bound tree: $p = p + 1$.
- Determine the minimal delaying set DS which contains the minimal delaying alternatives DA:

  $DS = \{DA | DA \subset S(t)$ and $\forall k \leq K$: $\sum_{i \in S(t)\backslash DA} r_{ik} \leq a_k$ and $\neg\exists j \in DA | \sum_{i \in S(t)\backslash DA} r_{ik} + r_{jk} \leq a_k \}$.

  Determine the corresponding set of delaying modes MS which contains the delaying modes DM:
  $MS = \{DM | DM = (k \prec DA)$, $k \in S(t) \backslash DA$ and $DA \in DS\}$.
- Delete all minimal delaying modes satisfying the subset dominance rule, i.e. $MS = MS \backslash \{DM | AC_y \subset (AC_x \cup DM)\}$ with y a previously examined node in the branch-and-bound tree.
- Compute for each non-dominated delaying mode a lower bound lb on the weighted earliness-tardiness cost using the recursive solution procedure.
- Delete all minimal delaying modes for which $lb \geq ub$, i.e. $MS = MS \backslash \{DM | lb \geq ub\}$.
- Go to STEP 3.

STEP 3. RESOURCE ANALYSIS

- Do for each non-dominated delaying mode :
  - Determine the first period in the optimal schedule in which a resource conflict occurs, i.e. the first period $]t\text{-}1,t]$ for which $\exists k \le K$: $\sum_{i \in S(t)} r_{ik} \le a_k$.
  - If there is no resource conflict and $lb < ub$, update $ub = lb$.
- Go to STEP 4.

STEP 4. BRANCHING

- If there are no delaying modes left at this level p with $lb < ub$, go to STEP 5.
- Select the delaying mode $DM \in MS$ with the smallest lb and add the additional precedence relations, i.e. $AC_x = AC_x \cup DM$.
- Go to STEP 2.

STEP 5. BACKTRACKING

- Delete the additional precedence relations inserted at level p, i.e. $AC_x = AC_x \backslash DM$.
- Decrease the level of the branch-and-bound tree: $p = p$ - 1.
- If the branching level $p > 0$, go to STEP 4 else STOP.

4. An example

In this section we will compute the optimal solution by means of an instance adapted from the Patterson set (Patterson 1984). The corresponding AON project network is shown in Figure 1. There are 7 activities (and two dummy activities) and one resource type with an availability of 5. The number above the node denotes the activity duration, while the numbers below the node denote the due date, the unit penalty cost (the unit earliness costs equals the unit tardiness costs) and the resource requirements, respectively.

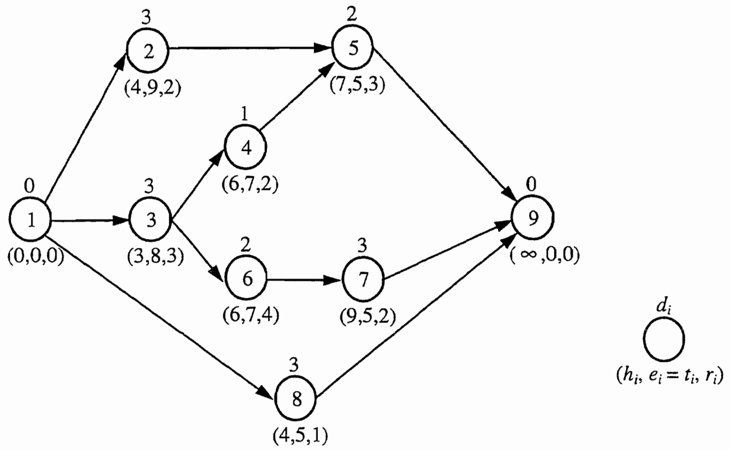

Figure 1. A project network from the Patterson set

The branch-and-bound tree for the example is given in Figure 2. At the initial level p = 0, the value of the optimal solution using the recursive procedure is lb = 5 with finishing times $f_2 = 4, f_3 = 3, f_4 = 6, f_5 = 8, f_6 = 6, f_7 = 9$ and $f_8 = 4$. Since there is a resource conflict at time instant 1 caused by activities 2, 3 and 8, we determine the minimal delaying set at the next level of the branch-and-bound tree. DS = {{2},{3},{8}} and therefore, we create 6 delaying modes (with six additional precedence relations) corresponding to the six nodes at level $p = 1$. We compute for each node a lower bound by the recursive procedure for the unconstrained weighted earliness-tardiness project scheduling problem. Now we select the delaying mode with the smallest lower bound, i.e. node 3 with $lb = 15$ and finishing times $f_2 = 4, f_3 = 3, f_4 = 6, f_5 = 8, f_6 = 6, f_7 = 9$ and $f_8 = 6$. Activities 4, 6 and 8 cause a resource conflict at time instant 5. We generate DS = {{4},{6}} and 4 delaying modes corresponding to nodes 8, 9, 10 and 11 at the level $p = 2$. Remark that the solution of node 9 is resource feasible. We update the current upper bound $ub = 27$. We continue with the delaying mode with the smallest lower bound, i.e. node 8 with $lb = 22$ and finishing times $f_2 = 4, f_3 = 3, f_4 = 6, f_5 = 8, f_6 = 5, f_7 = 9$ and $f_8 = 6$. Since the resource requirements at time instant 3 exceed the resource availibilities, the solution of this node is not resource feasible either. Again, we generate DS = {{2},{6}} and 4 delaying modes corresponding to nodes 12, 13, 14 and 15 at level $p = 3$. Node 12 can be fathomed by node 10 due to the subset dominance rule (this is denoted by D_{10} in Figure 2). We compute a lower bound for nodes 13, 14 and 15. All three nodes can be fathomed because the lower bound is greater than the current upper bound. The procedure now backtracks to level $p = 2$ and selects node 11 which has the smallest lower bound. The algorithm continues this way until it returns at the initial node at level 0. The optimal solution of the example has a weighted earliness-tardiness cost of 27 as shown in node 9 of Figure 2.

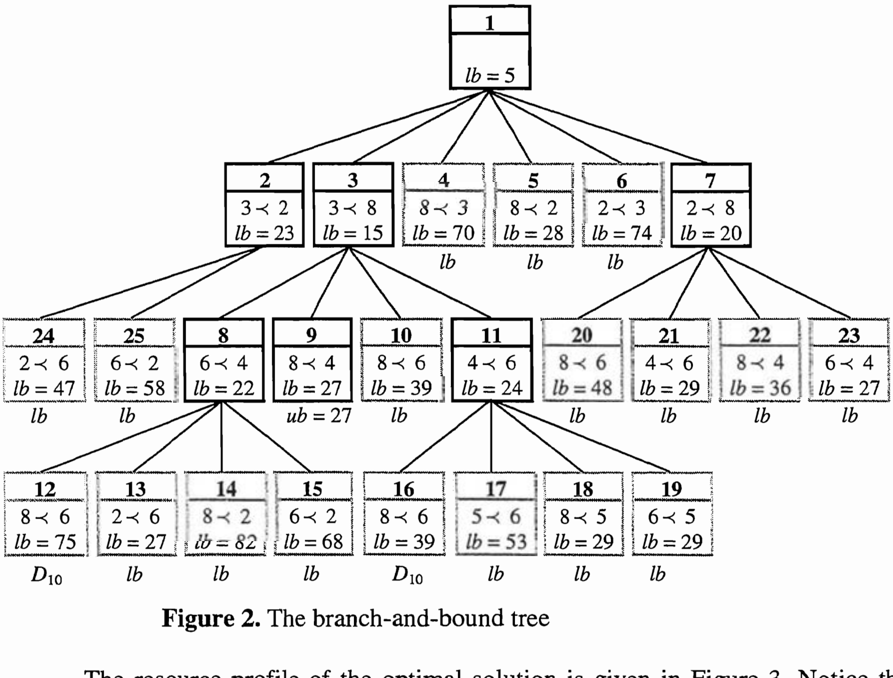

Figure 2. The branch-and-bound tree

The resource profile of the optimal solution is given in Figure 3. Notice that an optimal solution with respect to the makespan objective does not necessarily correspond to the optimal solution of the *RCPSPWET*. The minimal makespan of this example is 8 while the optimal weighted earliness-tardiness-schedule corresponds to a makespan of 9.

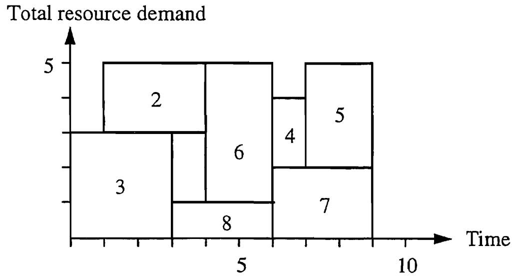

Figure 3. The optimal resource profile

5. Computational experience

The branch-and-bound algorithm has been coded in Visual C++ Version 4.0 under Windows NT 4.0 on a Dell personal computer (Pentium 200 MHz processor). In order to validate our branch-and-bound algorithm, we used 7,560 problem instances generated by *ProGen/Max* (Schwindt, 1995). The settings of these instances in activity-on-the-node format are described in Table I. We obtained 756 problem classes, each consisting of 10 instances. The problem set was extended with unit penalty costs for each activity randomly generated between 1 and 10.

The due dates were generated as follows: first, we obtained a maximum due date for each project by multiplying the critical path length with a factor as given in table I. Then we randomly generated numbers between 1 and the maximum due date. Finally, we sorted these numbers and assigned them to the activities in increasing order, i.e. activity 1 has the lowest due date, activity 2 the second lowest, etc..

Table I. Parameter settings used to generate the test instances

number of activities	10, 20 or 30
order strength (*OS*) *(Mastor, 1970)*	0.25, 0.50 or 0.75
resource factor (*RF*) *(Pascoe, 1966)*	0.25, 0.50, 0.75 or 1.00
resource strength (*RS*) *(Kolisch et al., 1995)*	0.00, 0.25 or 0.50
due dates of the activities	randomly selected with factor 1.00, 1.25, 1.50, 1.75, 2.00, 2.25 or 2.50

Table II represents the average CPU-time and its standard deviation in seconds for a varying number of activities and a time limit of 100 seconds. The number of activities has a significant effect on the average CPU-time and on the number of problems solved to optimality, as displayed in Table II and Figure 4 respectively. All problems with 10 activities can be solved to optimality within 1 second of CPU-time. For problems containing 20 activities, 93.8% of the number of problems can be solved to optimality when the allowed CPU-time is 1 second, whereas 99.8% of the number of problems can be solved when the time limit is 100 seconds. For problems with 30 activities, 79.4% of the number of problems can be solved within 1 second of CPU-time whereas 90.8% of the number of problems can be solved to optimality when the allowed CPU-time is 100 seconds.

Table II. Impact of the number of activities

# activities	# problems	Average CPU-time	Standard Deviation
10	2520	0.001	0.006
20	2520	0.508	4.128
30	2520	11.391	29.962

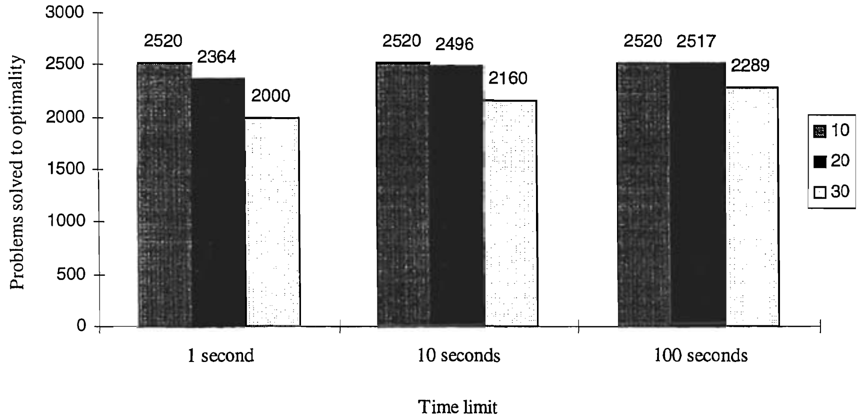

Figure 4. Effect of the number of activities

The effect of *OS* on the number of problems solved to optimality is displayed in Figure 5. As expected, the order strength has a negative correlation with the problem hardness, that is, the higher *OS*, the easier the problem. In Vanhoucke et al. (1999) we have observed that for the *WETPSP* the opposite is true. That means that, although *OS* has a positive correlation with the problem hardness for the *WETPSP*, the overall effect for the *RCPSPWET* remains negative.

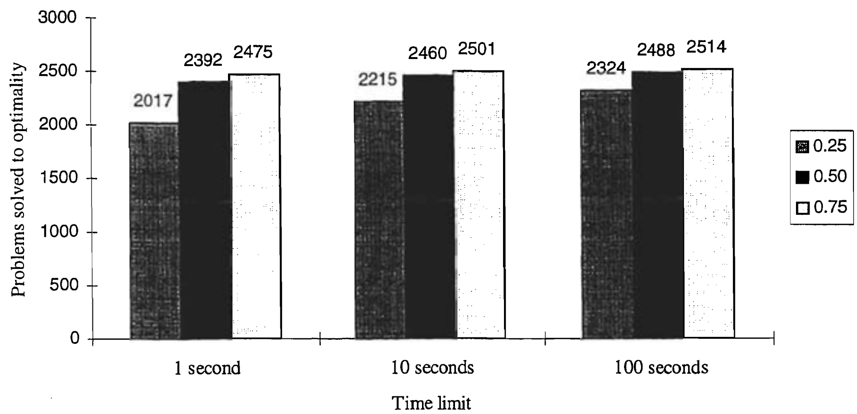

Figure 5. Effect of the order strength (*OS*)

Figures 6 and 7 display the effect of the resource factor (*RF*) and the resource strength (*RS*), respectively, on the number of problems solved to optimality within an allocated CPU-time. The higher the resource factor, the more difficult the instances. An opposite effect can be observed for the resource strength. The number of problems solved to optimality increases when *RS* increases. These results were also observed by Kolisch et al. (1995) and De Reyck and Herroelen (1996).

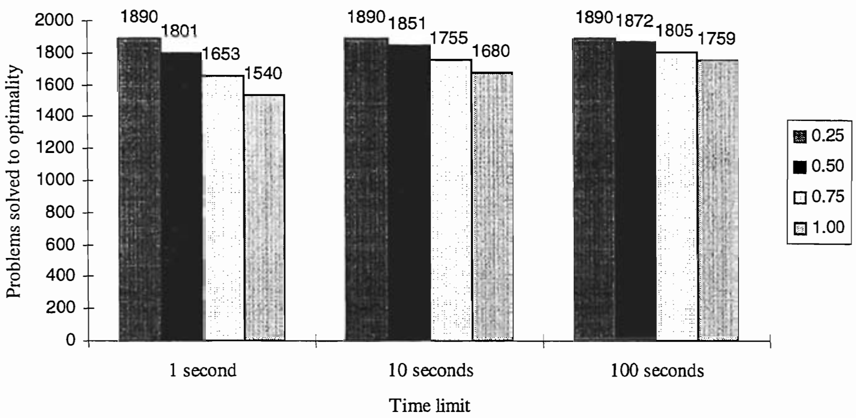

Figure 6. Effect of the resource factor (*RF*)

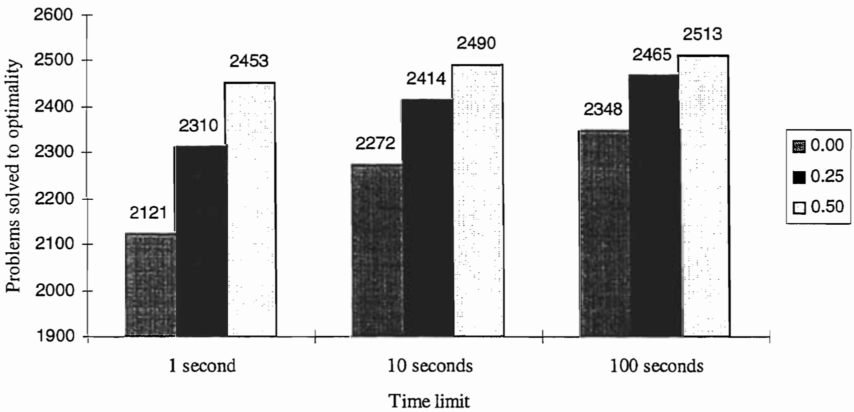

Figure 7. Effect of the resource strength (*RS*)

Table III illustrates the effect of the due date for a different number of activities on the CPU time with a time limit of 100 seconds. The negative correlation between the due date factor and the hardness of the problem is due to two reasons. First, this effect was also observed for the *WETPSP*: when the factor used for the due date generation is small, the problem contains many binding precedence relations and an extensive search will be needed to shift a large number of sets of activities to solve the problem. Problems with a large factor for the due date generation contain only few binding precedence relations in the due date tree. In that case, many activities will be scheduled on their due date and only a small number of shifts will be needed to solve the problem. Second, when the due date factor is large, the number of nodes in the search tree will decrease dramatically since the probability of a resource conflict will decrease. Both the number of nodes in the search tree and the time spent per node are negatively correlated with the due date factor.

Table III. Impact of due date factor on the CPU-time

	1.00	1.25	1.50	1.75	2.00	2.25	2.50
10 activities	0.002	0.002	0.001	0.001	0.001	0.001	0.001
20 activities	1.221	1.103	0.723	0.212	0.139	0.107	0.05
30 activities	22.684	17.135	12.508	9.828	7.651	5.455	4.475

Table IV shows the effect of the subset dominance rule on the average number of nodes in the search tree without exceeding a time limit of 100 seconds. In each node a call to the recursive search algorithm is performed to solve the *WETPSP*. The table reveals that the subset dominance rule reduces the number of subproblems solved during the search process with, on the average, 14%.

Table IV. Impact of the subset dominance rule on the number of subproblems solved

	with dominance rule	without dominance rule
10 activities	78	89
20 activities	20,992	23,813
30 activities	355,358	414,560

6. Conclusions

In this paper, we presented a branch-and-bound procedure for the resource-constrained project scheduling problem with weighted earliness-tardiness penalty costs (*RCPSPWET*; $m,1|cpm|early/tardy$) based on a fast exact recursive search algorithm for the unconstrained weighted earliness-tardiness problem (*WETPSP*; $cpm|early/tardy$). Activities have a known deterministic due date, a unit earliness and unit tardiness penalty cost and renewable resource requirements. The objective is to schedule the activities in order to minimize the total weighted earliness-tardiness penalty cost subject to both the precedence and resource constraints. To the best of our knowledge, our procedure is the first exact algorithm for solving the *RCPSPWET*.

The branching strategy of the depth-first branch-and-bound algorithm makes use of a fast recursive search algorithm for the unconstrained weighted earliness-tardiness problem to compute the lower bounds. The resource conflicts are solved by generating minimal delaying alternatives and by introducing extra precedence relations. A subset dominance rule is used for additional node fathoming.

The procedure has been coded in Visual C++, version 4.0 under Windows NT and has been validated on a set of 7,560 problem instances, randomly generated using *ProGen/Max* (Schwindt, 1995). The results of the extensive computational tests obtained on a Dell personal computer (Pentium 200 MHz) are rather encouraging. Although the number of activities is found to have a significant effect on the required average CPU-time and the number of problems solved to optimality, the procedure is rather robust. Problems with 30 activities can be solved in an average CPU-time of 11 seconds (some 80% of the problems can be solved within 1 second of CPU-time; while this percentage goes up to 90% when the CPU-time allowance is increased to 100 seconds). An investigation of the impact of the topological structure of the network reveals that problems become easier to solve as the order strength goes up. The higher the resource factor, the more difficult the problem. An opposite effect has been observed for the resource strength. The tighter the due date, the more difficult the problem. The subset dominance rule allowed to fathom on the average 14% of the nodes in the search tree. The procedure holds the promise to handle other types of nonregular objective functions (such as the maximization of the net present value of the project) with similar performance.

References

De Reyck, B., 1998, "Scheduling projects with generalized precedence relations, exact and heuristic procedures", *Ph.D. Dissertation*, Katholieke Universiteit Leuven.

De Reyck, B. and Herroelen, W., 1996, "A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations", *European Journal of Operational Research*, 111, 125-174.

De Reyck, B. and Herroelen, W., 1998, "An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations", *Computers and Operations Research*, 25, 1-17.

Demeulemeester, E. and Herroelen, W., 1992, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science*, 38, 1803-1818.

Demeulemeester, E. and Herroelen, W., 1997, "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, 43, 1485-1492.

Herroelen, W., Van Dommelen, P. and Demeulemeester, E., 1997, "Project network models with discounted cash flows: A guided tour through recent developments", *European Journal of Operational Research*, 100, 97-121.

Herroelen, W., Demeulemeester, E. and De Reyck, B., 1998, "A classification scheme for project scheduling problems", in: Weglarz J. (Ed.), *Handbook on Recent advances in Project Scheduling*, Kluwer Academic Publishers, to appear.

Icmeli, O. and Erengüç, S.S., 1996, "A branch-and-bound procedure for the resource-constrained project scheduling problem with discounted cash flows", *Management Science*, 42, 1395-1408.

Kolisch, R., Sprecher, A. and Drexl, A., 1995, "Characterization and generation of a general class of resource-constrained project scheduling problems", *Management Science*, 41, 1693-1703.

Mastor, A.A., 1970, "An experimental and comparative evaluation of production line balancing techniques", *Management Science*, 16, 728-746.

Pascoe, T.L., 1966, "Allocation of resources - CPM", *Revue Française de Recherche Opérationelle*, 38, 31-38.

Patterson, J.H., 1984, "A comparison of exact procedures for solving the multiple resource-constrained project scheduling problem", *Management Science*, 30, 854-867.

Schwindt, C., 1995, "A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags" WIOR-Report-449, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe.

Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 1999, "An exact procedure for the unconstrained weighted earliness-tardiness project scheduling problem", Research Report 9907, Department of Applied Economics, Katholieke Universiteit Leuven.